Creating efficient Docker containers for FV3GFS

Mark Cheeseman, Johan Dahm, Eddie Davis, Oliver Elbert, Oliver Fuhrer, Rhea George, Jeremy McGibbon, Tobias Wicky
Vulcan Inc

UFS Users Workshop
July 29, 2020

VULCAN

Vulcan Climate Modeling

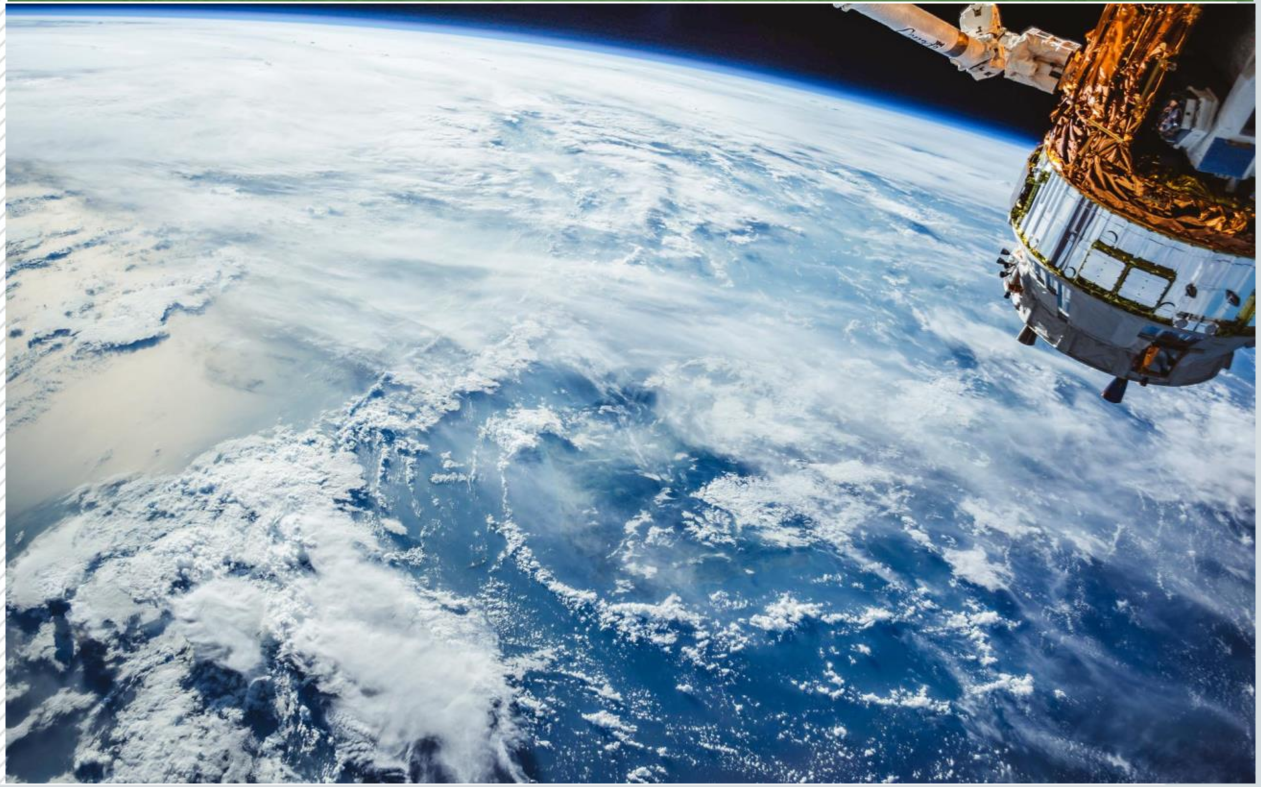
Group formed a little over one year ago

Consists of two teams (DSL, ML) with a total of 15 people based in Seattle and Princeton.

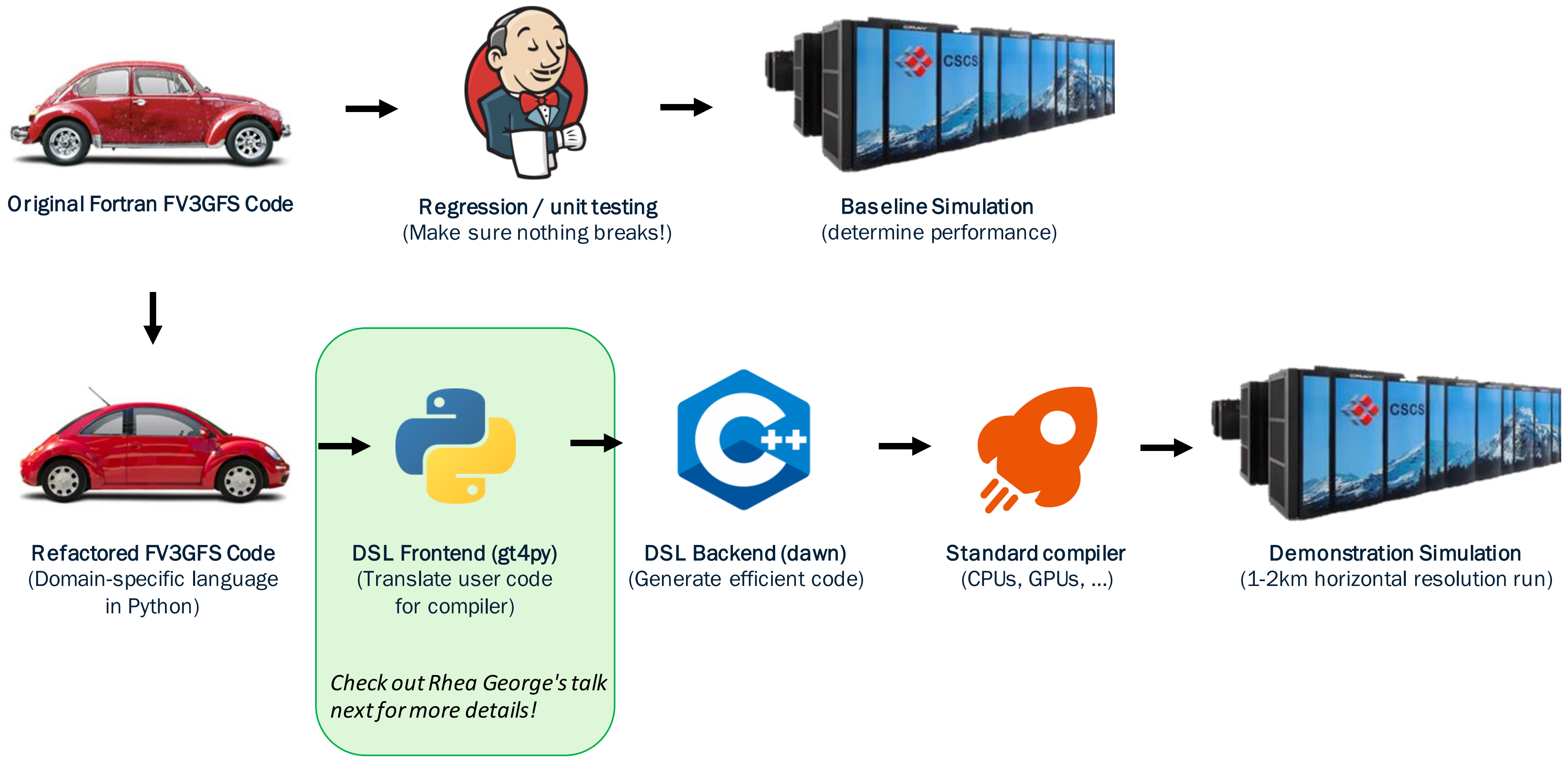
Led by Oliver Fuhrer and Chris Bretherton

Mission goals

1. *Improve the FV3GFS model to allow global storm-resolving simulations*
2. *Improve sub-grid cloud and precipitation parameterizations using ML-training on global cloud resolving model output.*



Domain Specific Language Team



Original Fortran FV3GFS Code

Regression / unit testing
(Make sure nothing breaks!)

Baseline Simulation
(determine performance)

Refactored FV3GFS Code
(Domain-specific language in Python)

DSL Frontend (gt4py)
(Translate user code for compiler)

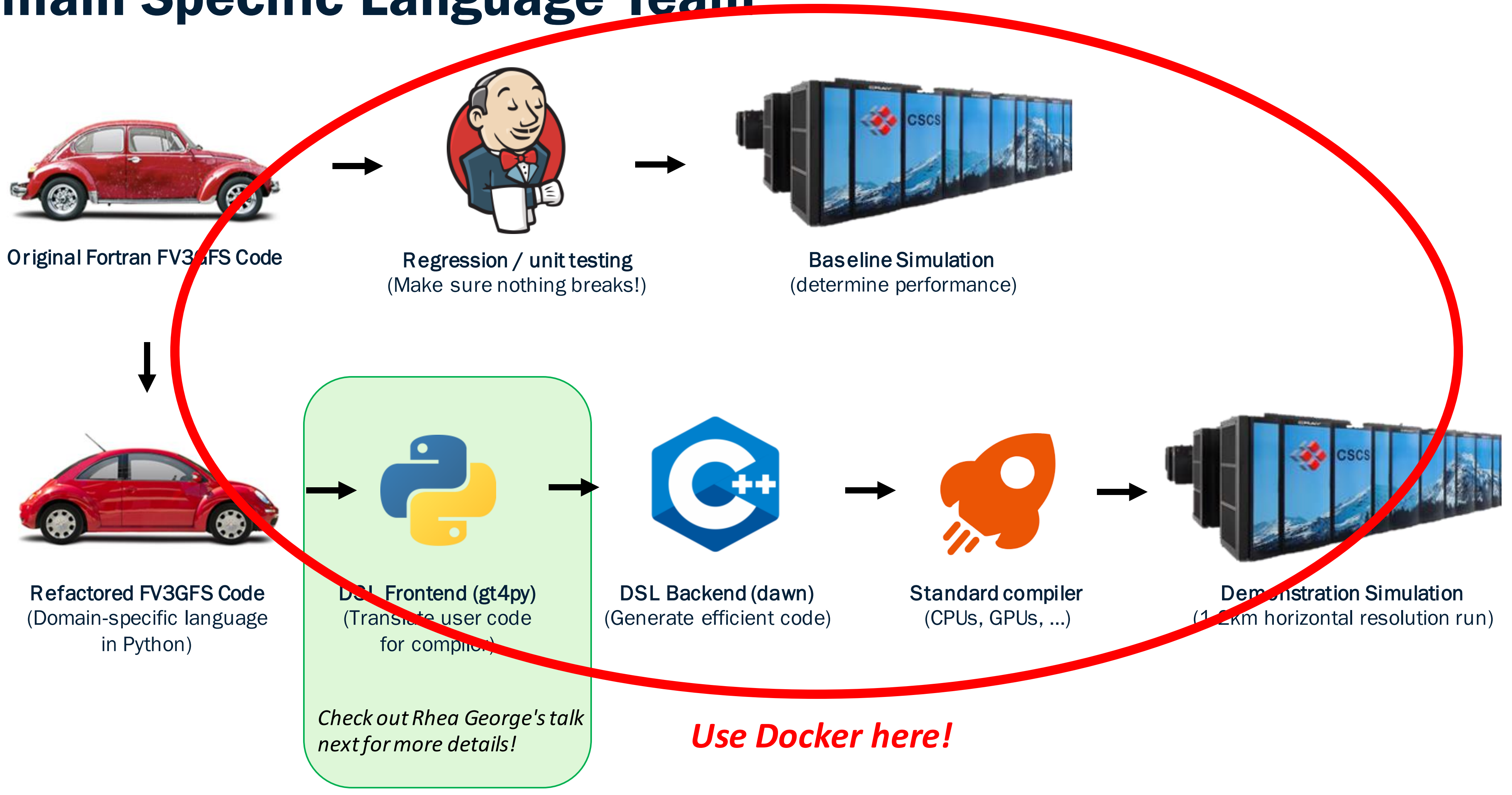
Check out Rhea George's talk next for more details!

DSL Backend (dawn)
(Generate efficient code)

Standard compiler
(CPUs, GPUs, ...)

Demonstration Simulation
(1-2km horizontal resolution run)

Domain Specific Language Team


Original Fortran FV3GFS Code
Regression / unit testing
(Make sure nothing breaks!)
Baseline Simulation
(determine performance)
Refactored FV3GFS Code
(Domain-specific language in Python)
DSL Frontend (gt4py)
(Translate user code for compiler)
Check out Rhea George's talk next for more details!
DSL Backend (dawn)
(Generate efficient code)
Standard compiler
(CPUs, GPUs, ...)
Demonstration Simulation
(1-2km horizontal resolution run)
Use Docker here!

Why use Docker?

It's a pain in the butt re-compiling on different platforms

Need to know where compilers, libraries are installed.

May need to build multiple prerequisite libraries: NetCDF, ESMF, FMS, etc.

What if the machine doesn't have your default/desired compiler?

Lots of Python packages: we have a current Python build requiring 28 individual packages

Can build the container locally or on a more "friendly" environment like a cloud VM.

Portability

Ideally, just download a Docker container and run

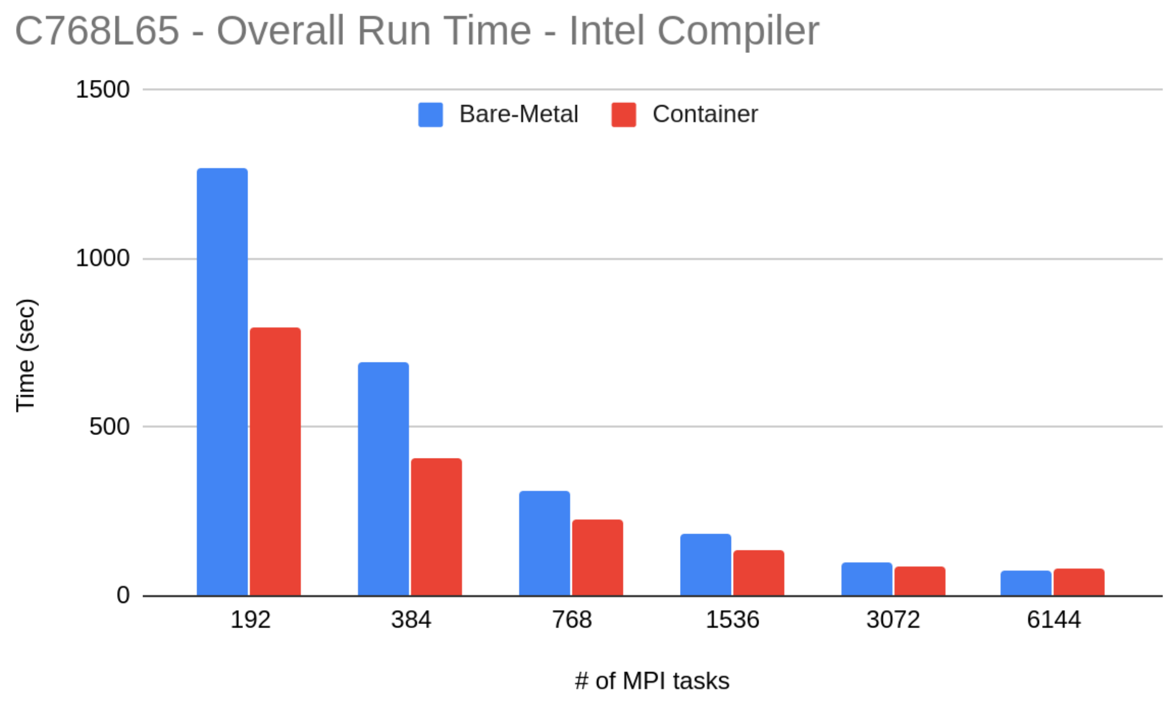
Only platform-specific info needed is that needed in running the container

Quicker to get running on different platforms (we run on local laptops, Google Cloud and CSCS' Cray supercomputer)

Why use Docker?

Docker is slow on HPC, right?

- *Achieving bare-metal performance is possible*
- *Easy MPI (and accelerator) support is available*
- *Can sometimes be more efficient than running bare-metal (I/O caching)*
    - *See first columns in plot*
    - *I/O caching and communication optimzation (dur to running on larger node) leads to significantly lower run time*



* GFS_v15.2.1, 2 model hour run, no restarts
* runs performed on the Piz Daint supercomputer at CSCS
* bare-metal runs on GPU partition of Piz Daint with 12 core nodes
* Container runs on multicore partition of Piz Daint with 36 core nodes

Running on Cray

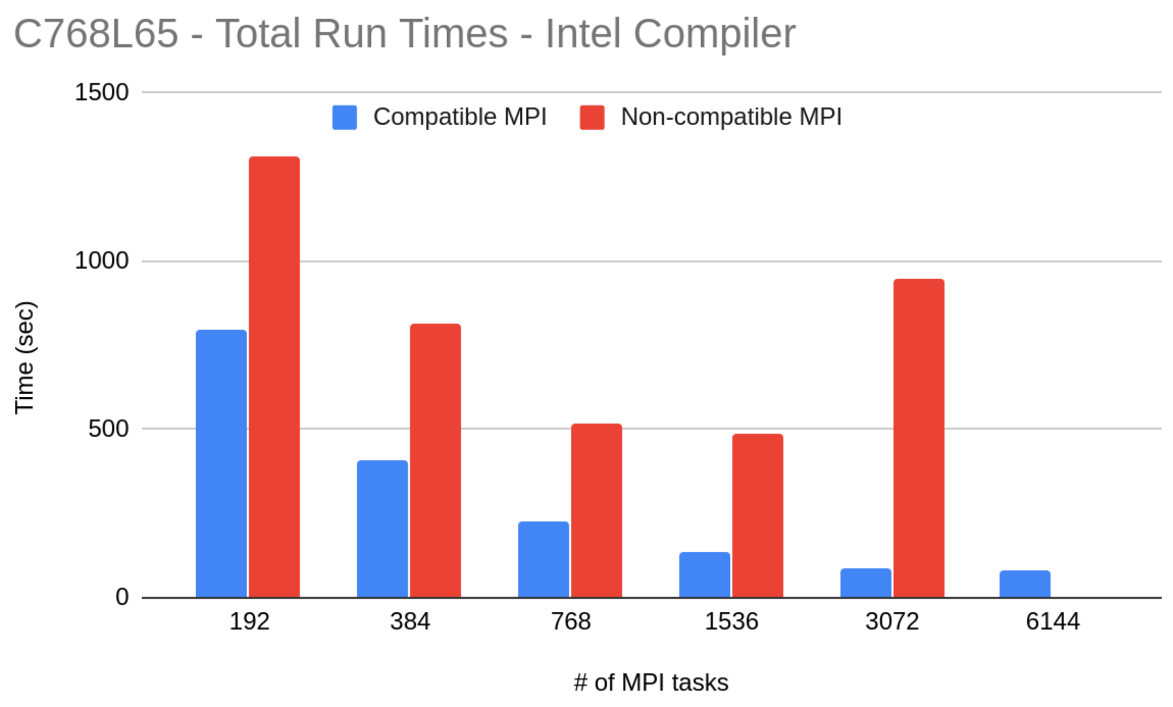
Runs performed on the Cray XC-50 "Piz Daint" located at the Swiss Supercomputing Center.

Support on HPC via tools like Sarus and Singularity

Optimal MPI performance requires some constraints in how we build FV3GFS.

Need to use a specifc MPI implementation & version (eg. MPICH 3.1.4). Not using this version results in sub-optimal communication performance, or FV3GFS not running at all.

[*https://sarus.readthedocs.io/en/stable/](https://sarus.readthedocs.io/en/stable/) **
https://sylabs.io/singularity/



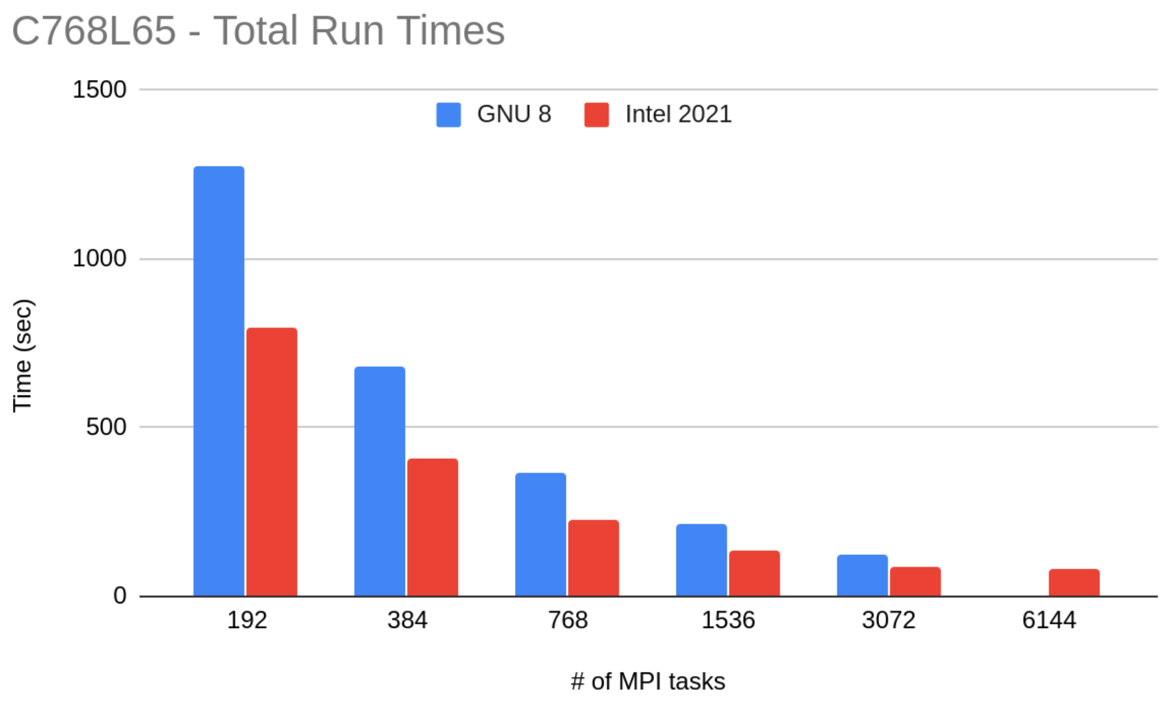
* GFS_v15.2.1, 2 model hour run, no restarts
* runs performed on the Piz Daint supercomputer at CSCS
* all runs performed on multicore partition of Piz Daint with 36 core nodes

Compiler support

Intel compiled-binary leads to the best performance (bare-metal and in container). Using Intel does lead to some issues:

- *Licensing (unless you use the free oneAPI compilers)*
- *Large-ish image sizes. oneAPI HPC Toolkit image* is ~5GB. Makes for more awkward transfers and image builds.*
- *Latest Intel MPI not compatible with Cray network. Need to build MPICH in image and re-configure.*
- *Python-Fortran binding code required different linking libraries for GNU and Intel builds.*

*https://hub.docker.com/r/intel/oneapi-hpckit



* GFS_v15.2.1, 2 model hour run, no restarts
* runs performed on the Piz Daint supercomputer at CSCS
* all runs performed on multicore partition of Piz Daint with 36 core nodes

Building Docker Images

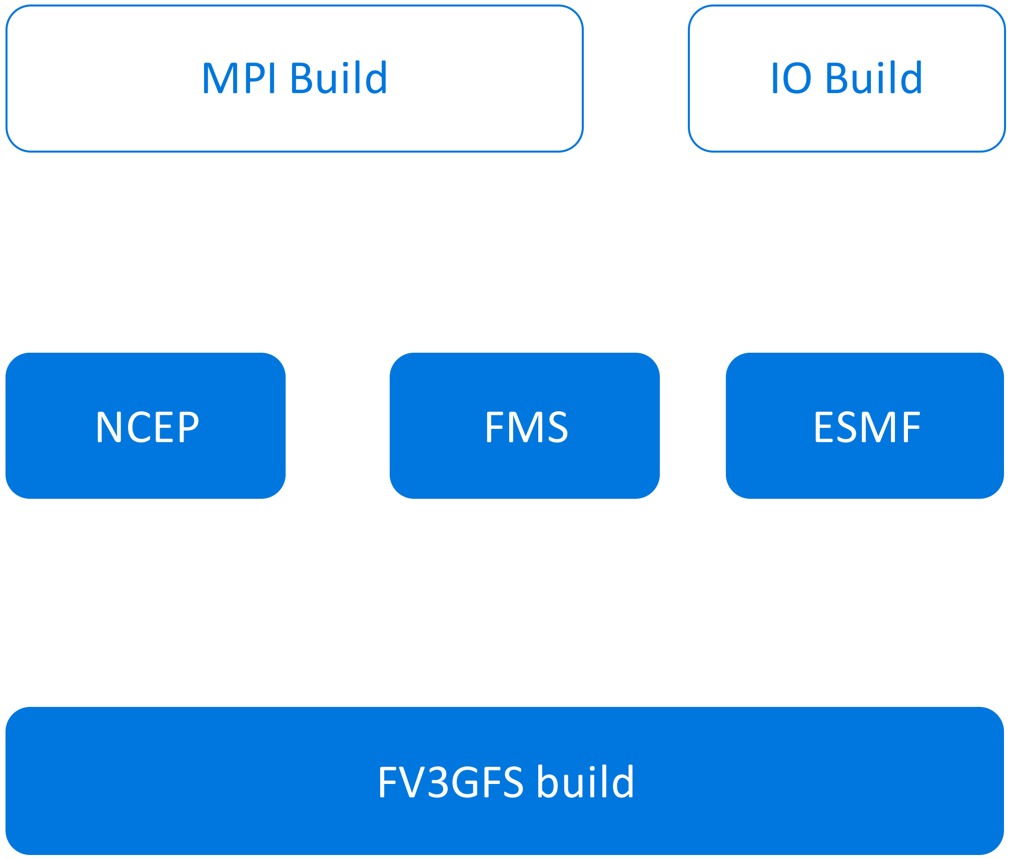
Use Docker Buildkit*

Building multiple stages in parallel to save build time. Eg, FMS and ESMF concurrent builds. Can also reduce the final image size by not loading libraries need for intermediate builds.

Keep images small

Don't include input data. Think smaller, agile containers instead of one large monolithic one. Makes transferring them easier.

*https://docs.docker.com/develop/develop-images/build_enhancements/



Building Docker Images

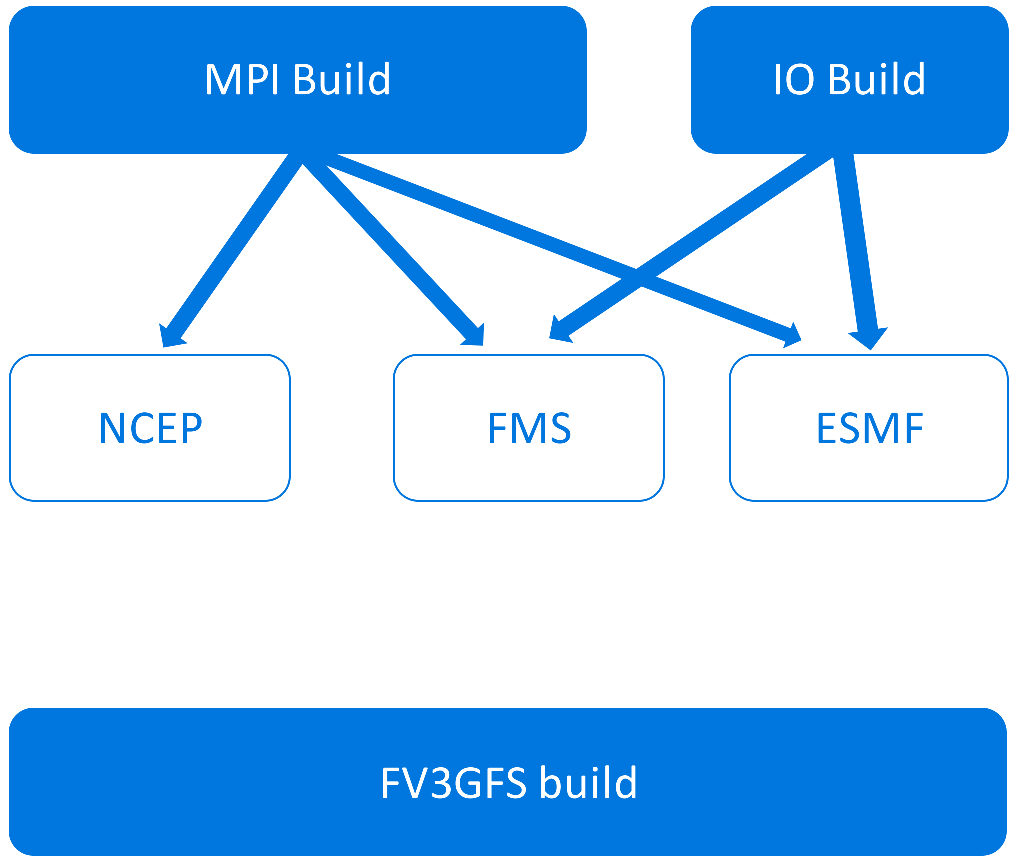
Use Docker Buildkit*

Building multiple stages in parallel to save build time. Eg, FMS and ESMF concurrent builds. Can also reduce the final image size by not loading libraries need for intermediate builds.

Keep images small

Don't include input data. Think smaller, agile containers instead of one large monolithic one. Makes transferring them easier.

*https://docs.docker.com/develop/develop-images/build_enhancements/



Building Docker Images

Use Docker Buildkit*

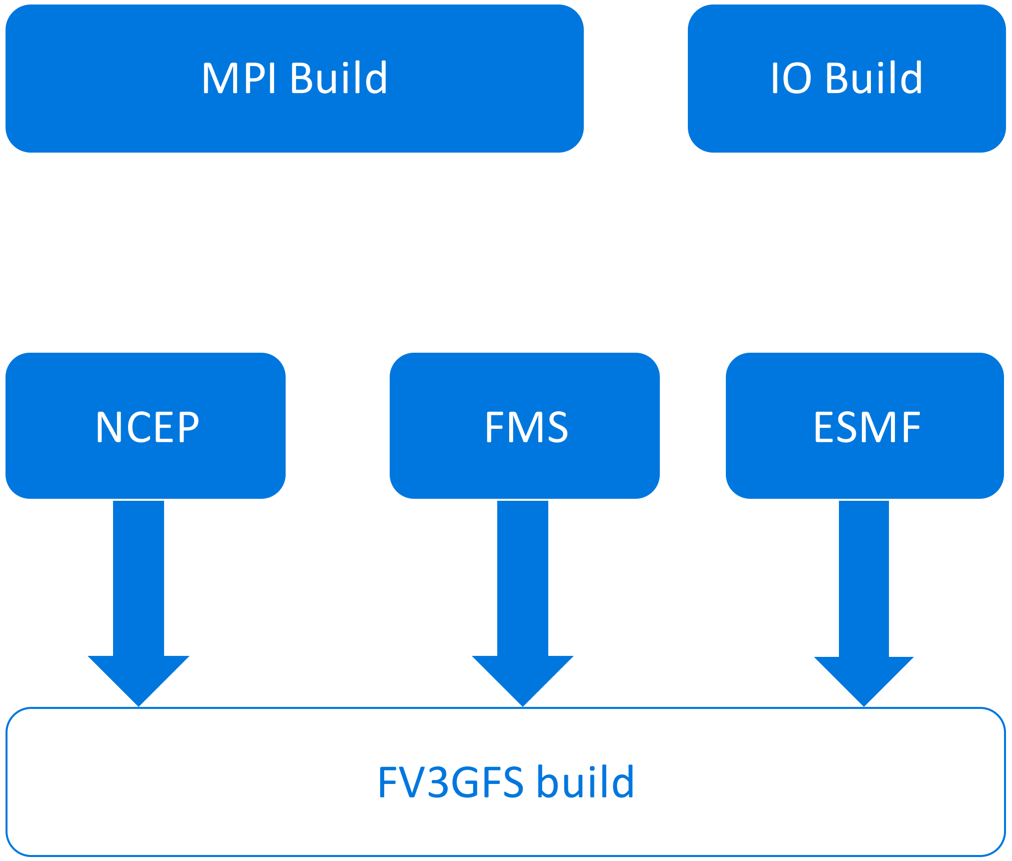
Building multiple stages in parallel to save build time. Eg, FMS and ESMF concurrent builds. Can also reduce the final image size by not loading libraries need for intermediate builds.

We save ~30 minutes for a full Intel build of FV3GFS using Docker Buildkit.

Keep images small

Don't include input data. Think smaller, agile containers instead of one large monolithic one. Makes transferring them easier.

*https://docs.docker.com/develop/develop-images/build_enhancements/



Wrapping Up

Docker containers are a viable deployment choice for running FV3GFS

- *Tools exisit to allow deployment on today's supercomputers*
- *Same Docker container can run on different platforms (local HPC, cloud)*
- *Easy integration into CI/testing processes*

Vulcan Climate will be releasing public versions of our Docker containerized refactored FV3GFS

- *Including Dockerfiles, multiple compiler support, Cray-ready versions*

Thanks to our collaborators!

- *Especially CSCS for time on Piz Daint and GFDL for assistance in setup of simulations*